

## Assignment 3

---

### Title

Simulate a cloud scenario using CloudSim and run a scheduling algorithm that is not present in CloudSim.

### Requirements

1. Java JDK and JRE
2. CloudSim archives (CloudSim4)
3. Eclipse IDE

### Theory

#### A) CloudSim

1. CloudSim is an open-source framework, which is used to simulate cloud computing infrastructure and services.
2. It is developed by the CLOUDS Lab organization and is written entirely in Java.
3. It is used for modelling and simulating a cloud computing environment as a means for evaluating a hypothesis prior to software development in order to reproduce tests and results.
4. If you were to deploy an application or a website on the cloud and wanted to test the services and load that your product can handle and also tune its performance to overcome bottlenecks before risking deployment, then such evaluations could be performed by simply coding a simulation of that environment with the help of various flexible and scalable classes provided by the CloudSim package, free of cost.

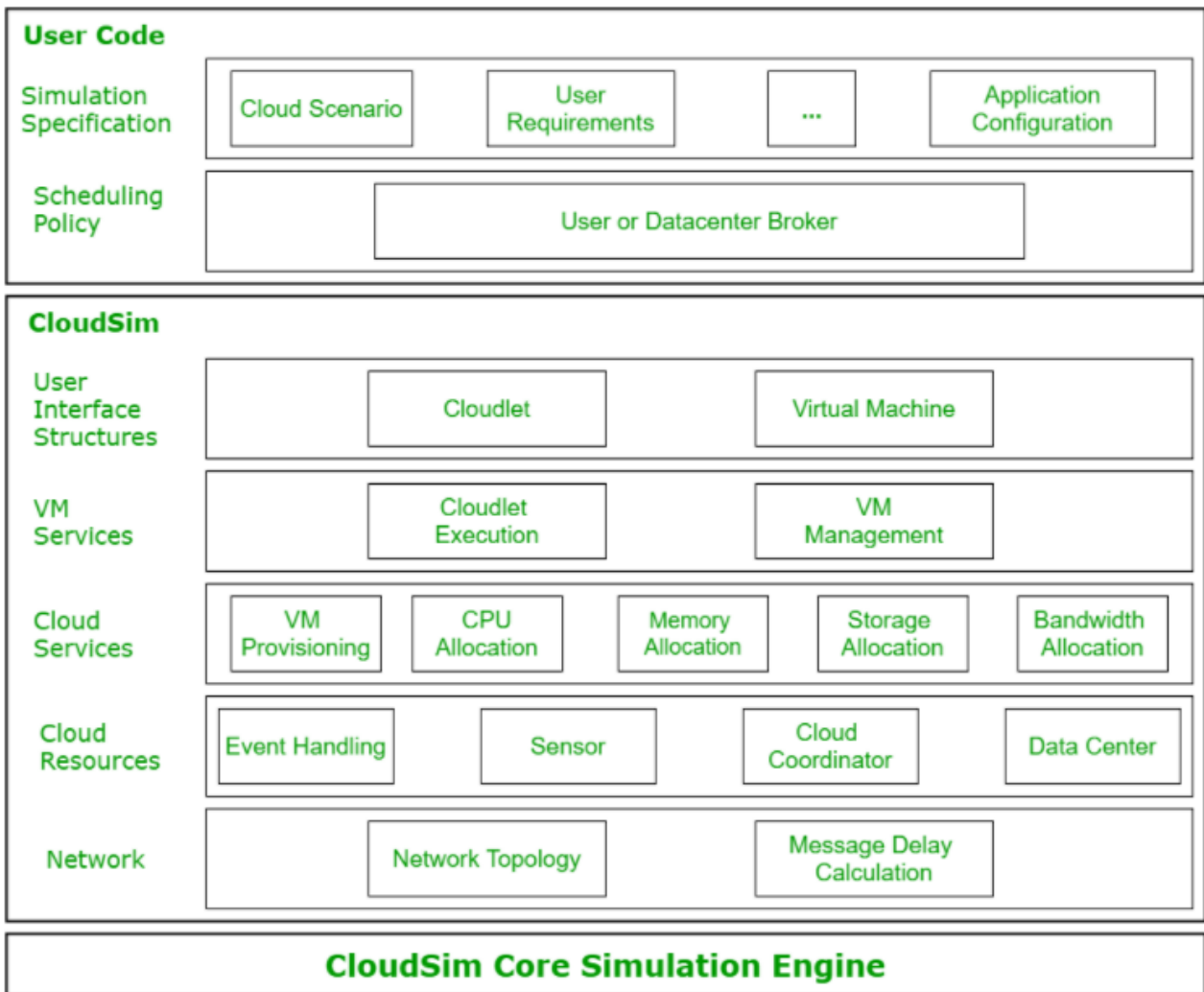
#### B) Benefits of CloudSim

1. No capital investment involved
2. Easy to use and Scalable
3. Risks can be evaluated at an earlier stage

4. No need for try-and-error approaches

### C) Architecture

1. CloudSim has a layered architecture which separates the User Code and the simulation environment.
2. It can be depicted as follows



*CloudSim Layered Architecture*

### D) CloudSim Components

- **Datacenter**: used for modelling the foundational hardware equipment of any cloud environment, that is the Datacenter. This class provides

methods to specify the functional requirements of the Datacenter as well as methods to set the allocation policies of the VMs etc.

- **Host:** this class executes actions related to management of virtual machines. It also defines policies for provisioning memory and bandwidth to the virtual machines, as well as allocating CPU cores to the virtual machines.
- **VM:** this class represents a virtual machine by providing data members defining a VM's bandwidth, RAM, mips (million instructions per second), size while also providing setter and getter methods for these parameters.
- **Cloudlet:** a cloudlet class represents any task that is run on a VM, like a processing task, or a memory access task, or a file updating task etc. It stores parameters defining the characteristics of a task such as its length, size, mi (million instructions) and provides methods similarly to VM class while also providing methods that define a task's execution time, status, cost and history.
- **DatacenterBroker:** is an entity acting on behalf of the user/customer. It is responsible for functioning of VMs, including VM creation, management, destruction and submission of cloudlets to the VM.
- **CloudSim:** this is the class responsible for initializing and starting the simulation environment after all the necessary cloud entities have been defined and later stopping after all the entities have been destroyed.

#### E) SJF algorithm

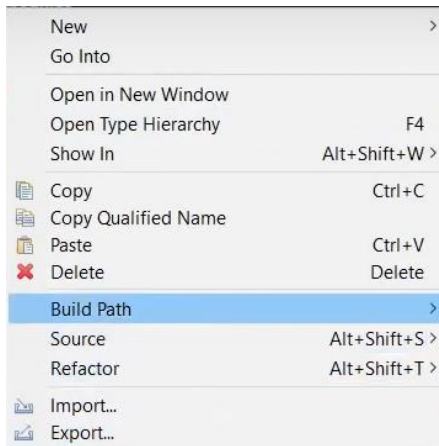
1. SJF stands for Shortest Job First
2. Shortest Job first has the advantage of having a minimum average waiting time among all scheduling algorithms.
3. It is a Greedy Algorithm.
4. It may cause starvation if shorter processes keep coming. This problem can be solved using the concept of ageing.
5. It is practically infeasible as Operating System may not know burst time and therefore may not sort them. While it is not possible to predict execution time, several methods can be used to estimate the execution time for a job, such as a weighted average of previous execution times.

SJF can be used in specialized environments where accurate estimates of running time are available.

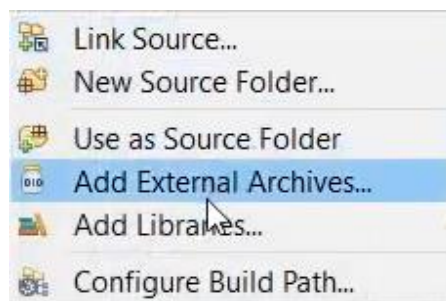
## Steps

### A) Installation of CloudSim and creation of simulation environment

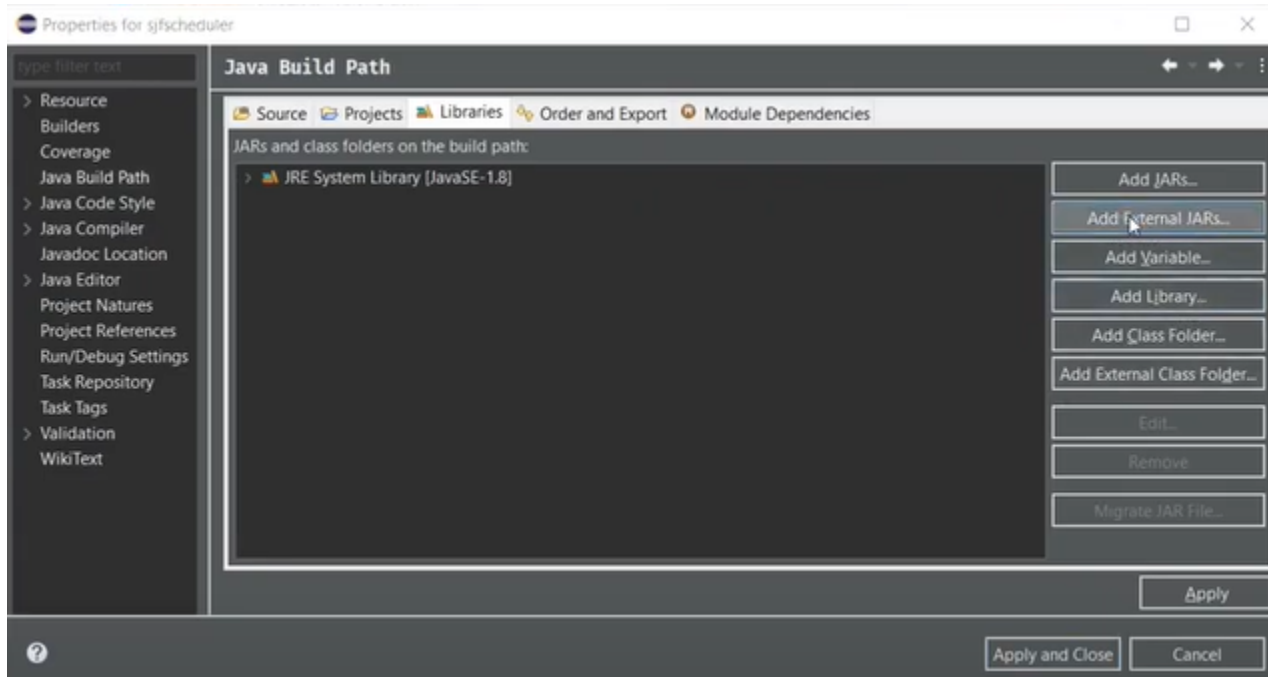
1. Visit <https://github.com/Cloudslab/cloudsim/releases> to download the CloudSim archives for CloudSim 4.
2. Extract the archive.
3. The jars folder of the extracted archive should contain the following files:
  - a) cloudsim-4.0.jar
  - b) cloudsim-examples.jar
4. Create a new Java Project using the Eclipse IDE.
5. Right click on the project root and select the Build Path option from the dropdown.



6. Select the Configure Build Path section from the extended dropdown



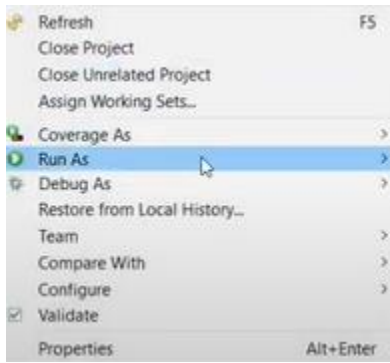
7. Select the Libraries section and click on Add External JARs field on the pop up



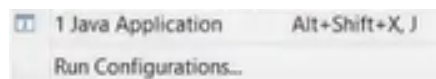
8. Navigate to the jars directory of the CloudSim archive and include the 2 jars in the project.
9. Create a new package in the src directory of the project.
10. Copy the code files for the constants, Data Center Creator, Data Center Broker, Matrix Generator and the SJF scheduler files from the <https://github.com/suyash-more/Cloud-Computing-Projects/tree/master/Scheduling-Algorithm-in-CloudSim/src> link.
11. Make sure that the package name provided in each file is the same as the previously created package.

## B) Execution of code

1. Right click on the project



2. Run the project as a Java Application (option in extended dropdown)



3. The result is displayed on the console

```
Starting SJF Scheduler...
Initializing new Matrices...
Initialising...
Starting CloudSim version 3.0
Datacenter_0 is starting...
Datacenter_1 is starting...
Datacenter_2 is starting...
Datacenter_3 is starting...
Datacenter_4 is starting...
Broker_0 is starting...
Entities started.
0.0: Broker_0: Cloud Resource List received with 5 resource(s)
0.0: Broker_0: Trying to Create VM #2 in Datacenter_0
0.0: Broker_0: Trying to Create VM #3 in Datacenter_1
0.0: Broker_0: Trying to Create VM #4 in Datacenter_2
0.0: Broker_0: Trying to Create VM #5 in Datacenter_3
0.0: Broker_0: Trying to Create VM #6 in Datacenter_4
0.1: Broker_0: VM #2 has been created in Datacenter #2, Host #0
0.1: Broker_0: VM #3 has been created in Datacenter #3, Host #0
0.1: Broker_0: VM #4 has been created in Datacenter #4, Host #0
0.1: Broker_0: VM #5 has been created in Datacenter #5, Host #0
0.1: Broker_0: VM #6 has been created in Datacenter #6, Host #0
0.1: Broker_0: Sending cloudlet 0 to VM #5
0.1: Broker_0: Sending cloudlet 1 to VM #6
0.1: Broker_0: Sending cloudlet 2 to VM #4
0.1: Broker_0: Sending cloudlet 3 to VM #3
0.1: Broker_0: Sending cloudlet 4 to VM #5
0.1: Broker_0: Sending cloudlet 5 to VM #2
0.1: Broker_0: Sending cloudlet 6 to VM #6
0.1: Broker_0: Sending cloudlet 7 to VM #3
0.1: Broker_0: Sending cloudlet 8 to VM #4
0.1: Broker_0: Sending cloudlet 9 to VM #4
0.1: Broker_0: Sending cloudlet 10 to VM #3
0.1: Broker_0: Sending cloudlet 11 to VM #2
0.1: Broker_0: Sending cloudlet 12 to VM #6
0.1: Broker_0: Sending cloudlet 13 to VM #4
```

```
1292.724: Broker_0: Cloudlet 0 received
1907.232: Broker_0: Cloudlet 5 received
2260.772: Broker_0: Cloudlet 1 received
2784.16: Broker_0: Cloudlet 3 received
2903.4: Broker_0: Cloudlet 2 received
3065.932: Broker_0: Cloudlet 11 received
4113.036: Broker_0: Cloudlet 4 received
4485.576: Broker_0: Cloudlet 18 received
4837.776: Broker_0: Cloudlet 20 received
4956.164: Broker_0: Cloudlet 6 received
5643.272: Broker_0: Cloudlet 8 received
5807.608: Broker_0: Cloudlet 21 received
6354.656: Broker_0: Cloudlet 7 received
6905.915999999999: Broker_0: Cloudlet 23 received
7719.535999999999: Broker_0: Cloudlet 24 received
8614.368: Broker_0: Cloudlet 12 received
8752.444: Broker_0: Cloudlet 10 received
8986.24: Broker_0: Cloudlet 9 received
10703.216: Broker_0: Cloudlet 15 received
10857.967999999999: Broker_0: Cloudlet 14 received
11948.996: Broker_0: Cloudlet 25 received
13310.556: Broker_0: Cloudlet 13 received
13635.776: Broker_0: Cloudlet 16 received
15582.328: Broker_0: Cloudlet 17 received
16230.772: Broker_0: Cloudlet 19 received
17007.956: Broker_0: Cloudlet 27 received
19003.152000000002: Broker_0: Cloudlet 22 received
19533.66: Broker_0: Cloudlet 28 received
22878.644: Broker_0: Cloudlet 26 received
25918.7: Broker_0: Cloudlet 29 received
25918.7: Broker_0: All Cloudlets executed. Finishing...
```



OUTPUT								
Cloudlet ID	STATUS	Data center ID	VM ID	Time	Start Time	Finish Time	Waiting Time	
00	SUCCESS	05	05	1292.62	00.1	1292.72	00	
05	SUCCESS	02	02	1907.13	00.1	1907.23	00	
01	SUCCESS	06	06	2260.67	00.1	2260.77	00	
03	SUCCESS	03	03	2784.06	00.1	2784.16	00	
02	SUCCESS	04	04	2903.3	00.1	2903.4	00	
11	SUCCESS	02	02	1158.7	1907.23	3065.93	1907.13	
04	SUCCESS	05	05	2820.31	1292.72	4113.04	1292.62	
18	SUCCESS	02	02	1419.64	3065.93	4485.58	3065.83	
20	SUCCESS	02	02	352.2	4485.58	4837.78	4485.48	
06	SUCCESS	06	06	2695.39	2260.77	4956.16	2260.67	
08	SUCCESS	04	04	2739.87	2903.4	5643.27	2903.3	
21	SUCCESS	05	05	1694.57	4113.04	5807.61	4112.94	
07	SUCCESS	03	03	3570.5	2784.16	6354.66	2784.06	
23	SUCCESS	02	02	2068.14	4837.78	6905.92	4837.68	
24	SUCCESS	02	02	813.62	6905.92	7719.54	6905.82	
12	SUCCESS	06	06	3658.2	4956.16	8614.37	4956.06	
10	SUCCESS	03	03	2397.79	6354.66	8752.44	6354.56	
09	SUCCESS	04	04	3342.97	5643.27	8986.24	5643.17	
15	SUCCESS	06	06	2088.85	8614.37	10703.22	8614.27	
14	SUCCESS	03	03	2105.52	8752.44	10857.97	8752.34	
25	SUCCESS	03	03	1091.03	10857.97	11949	10857.87	
13	SUCCESS	04	04	4324.32	8986.24	13310.56	8986.14	
16	SUCCESS	06	06	2932.56	10703.22	13635.78	10703.12	
17	SUCCESS	06	06	1946.55	13635.78	15582.33	13635.68	
19	SUCCESS	04	04	2920.22	13310.56	16230.77	13310.46	
27	SUCCESS	06	06	1425.63	15582.33	17007.96	15582.23	
22	SUCCESS	04	04	2772.38	16230.77	19003.15	16230.67	
28	SUCCESS	06	06	2525.7	17007.96	19533.66	17007.86	
26	SUCCESS	04	04	3875.49	19003.15	22878.64	19003.05	
29	SUCCESS	04	04	3040.06	22878.64	25918.7	22878.54	

Makespan using SJF: 4396.012266367984